# Optimal Bounds for Distinct Quartics

**Panagiotis Charalampopoulos**
School of Computing and Mathematical Sciences, Birkbeck, University of London, UK

**Paweł Gawrychowski**
Institute of Computer Science, University of Wrocław, Poland

**Samah Ghazawi**
Department of Computer Science, University of Haifa, Israel
Department of Software Engineering, Braude, College of Engineering, Karmiel, Israel

## Abstract

A fundamental concept related to strings is that of repetitions. It has been extensively studied in many versions, from both purely combinatorial and algorithmic angles. One of the most basic questions is how many distinct squares, i.e., distinct strings of the form $UU$, a string of length $n$ can contain as fragments. It turns out that this is always $\mathcal{O}(n)$, and the bound cannot be improved to sublinear in $n$ [Fraenkel and Simpson, JCTA 1998].

Several similar questions about repetitions in strings have been considered, and by now we seem to have a good understanding of their repetitive structure. For higher-dimensional strings, the basic concept of periodicity has been successfully extended and applied to design efficient algorithms—it is inherently more complex than for regular strings. Extending the notion of repetitions and understanding the repetitive structure of higher-dimensional strings is however far from complete.

Quartics were introduced by Apostolico and Brimkov [TCS 2000] as analogues of squares in two dimensions. Charalampopoulos, Radoszewski, Rytter, Waleń, and Zuba [ESA 2020] proved that the number of distinct quartics in an $n \times n$ 2D string is $\mathcal{O}(n^2 \log^2 n)$ and that they can be computed in $\mathcal{O}(n^2 \log^2 n)$ time. Gawrychowski, Ghazawi, and Landau [SPIRE 2021] constructed an infinite family of $n \times n$ 2D strings with $\Omega(n^2 \log n)$ distinct quartics. This brings the challenge of determining asymptotically tight bounds. Here, we settle both the combinatorial and the algorithmic aspects of this question: the number of distinct quartics in an $n \times n$ 2D string is $\mathcal{O}(n^2 \log n)$ and they can be computed in the worst-case optimal $\mathcal{O}(n^2 \log n)$ time.

As expected, our solution heavily exploits the periodic structure implied by occurrences of quartics. However, the two-dimensional nature of the problem introduces some technical challenges. Somewhat surprisingly, we overcome the final challenge for the combinatorial bound using a result of Marcus and Tardos [JCTA 2004] for permutation avoidance on matrices.

**2012 ACM Subject Classification** Theory of computation → Pattern matching

**Keywords and phrases** 2D strings, quartics, repetitions, periodicity

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2024.112

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: https://doi.org/10.48550/arXiv.2403.06667

## 1 Introduction

Repetitions are a staple topic of both combinatorics on words [22] and algorithms on strings [34]. In both areas, the classical objects of study are linear sequences of characters from a finite alphabet. Depending on whether we are more interested in their combinatorial properties or designing efficient algorithms for them, it is customary to call such sequences words or strings, respectively. In this paper, we use the latter convention.

Perhaps the most natural example of a repetition in a string is a square, that is, a string of the form $UU$, also known as a "tandem repeat" in the biological literature [54]. The basic





question concerning squares is whether any of the fragments of a string of length $n$ is a square, and, if so, what is the number of such fragments. The origins of this question can be traced back to Thue [76], who constructed an infinite string over a ternary alphabet that contains no squares. Thus, we can construct arbitrarily long square-free strings over such alphabets. The next question is what is the largest possible number of fragments that are squares. However, any even-length fragment of $\mathtt{a}^n$ is a square. One way to make the question non-trivial is to only consider the primitively rooted squares, meaning that $U$ is not a power of another string. This decreases the possible number of occurrences to $\mathcal{O}(n \log n)$, which is asymptotically tight [30]. Another way is to only consider distinct squares.

Fraenkel and Simpson [47] showed that any string of length $n$ contains at most $2n$ distinct squares and constructed an infinite family of strings such that each string $S$ in this family contains $|S| - \Theta(\sqrt{|S|})$ distinct squares. For many years, it was conjectured that the upper bound should be at most $n$. After a series of simplifications and improvements [42,58,59,66,75], the conjecture was finally proven by Brlek and Li [24], who showed an upper bound of $n - \sigma + 1$, where $\sigma$ is the size of the alphabet. The same authors [25] also showed an upper bound of $n - \Theta(\log n)$. On the algorithmic side, Apostolico and Preparata [16], Main and Lorentz [68] and Crochemore [31] showed how to find a compact representation of all squares (in particular, test square-freeness) in a string of length $n$ in $\mathcal{O}(n \log n)$ time. Specifically, such a representation stores all distinct squares. To obtain a faster algorithm for finding only the distinct squares, one needs to restrict the size of the alphabet. For constant alphabets, Gusfield and Stoye [55] designed an $\mathcal{O}(n)$ time algorithm. This was later generalized to the more general case of an integer alphabet (that can be sorted in linear time) [20, 37]. The complexity of testing square-freeness over general ordered and unordered alphabets of size $\sigma$ was very recently settled by Ellert and Fischer [44] providing a linear time algorithm, and Ellert et al. [45] providing an $\mathcal{O}(n \log \sigma)$ time algorithm, respectively; this problem has been also studied in the parallel [13, 14, 38, 39] and the online settings [56, 64, 65, 67]. Thus, by now we seem to have obtained a rather good understanding of both the combinatorial and the algorithmic properties of distinct squares. We stress that, while these properties are interesting on their own, and naturally the combinatorial bound was used to design efficient enumeration algorithms [21, 55], they were also crucial in designing efficient algorithms and data structures for other problems. For example, the Maximal Augmented Suffix Tree (MAST) introduced by Apostolico and Preparata [17] which enables counting the maximum number of non-overlapping occurrences uses $\mathcal{O}(n)$ space due to the linear upper bound on the number of distinct squares (as observed by Brodal et al. [26]). The same applies to the construction time and the space of the Cover Suffix Tree (CST) [62, 72].

Arguably, linear sequences are not always best suited to model the objects that we would like to study. A natural extension is to consider rectangular arrays of characters from a finite alphabet, which can be seen as 2D strings. Possible applications in image processing [73] sparked interest in designing algorithms for searching in 2D strings already in the late 1970s [18, 23]. This turned out to be significantly more challenging than searching in 1D strings: both versions were studied already in the 70s, but while for 1D strings an alphabet-independent linear-time algorithm had been soon found [61], achieving the same goal for 2D strings took till the 90s [3, 40, 48]. Extensions of this basic problem such as approximate searching [9, 29], indexing [28, 50, 51], searching in smaller space [33], scaled searching [6, 7], searching in random 2D strings [60], dictionary searching [8, 57, 70], and searching in compressed 2D strings [1, 4, 11] have been also considered.

The combinatorial structure of 2D strings seems to be significantly more involved than that of 1D strings. As a prime example, the basic tool used in algorithms and combinatorics on

1D strings is periodicity. We say that $p$ is a period of a 1D string $S[1\,..\,n]$ when $S[i] = S[i+p]$ for all $i = 1, 2, \ldots, n-p$. The set of all periods is very structured due to a classical result of Fine and Wilf [46] according to which for any two periods $p, q$ such that $p + q \leq n$, the greatest common divisor of $p$ and $q$ is also a period. The natural way to extend this notion to 2D strings is to define $(x, y)$ to be a period of a 2D string $S[1\,..\,n][1\,..\,n]$ when $S[i][j] = S[i + x][j + y]$ for all $i = 1, 2, \ldots, n - x$ and $j = 1, 2, \ldots, n - y$. This notion was introduced by Amir and Benson [2], who provided a detailed study based on classifying 2D strings into four periodicity classes. This classification was later crucial in designing solutions for pattern matching, namely, an alphabet-independent linear-time algorithms [40, 48] and alphabet-independent optimal parallel algorithms [5, 32].

The rich combinatorial structure of 2D strings brings the challenge of finding the right generalization of the concept of repetitions. Apostolico and Brimkov [12] introduced two notions of repetitions in 2D strings that can be seen as natural analogues of squares in 1D strings. A tandem $W^{1,2}$ (or $W^{2,1}$) consists of 2 occurrences of the same block $W$ arranged in a $1 \times 2$ (or $2 \times 1$) pattern. Next, a quartic $W^{2,2}$ consists of 4 occurrences of the same block $W$ arranged in a $2 \times 2$ pattern. Note that Apostolico and Brimkov [12] additionally required that $W$ is primitive, meaning that it cannot be partitioned into non-overlapping copies of another block. However, it is more natural to call such tandems and quartics primitively rooted, as in [27]. Apostolico and Brimkov [12] showed asymptotically tight bounds of $\mathcal{O}(n^3 \log n)$ and $\mathcal{O}(n^2 \log^2 n)$ for the number of primitively rooted tandems and quartics, respectively. The former bound was later complemented with a worst-case optimal $\mathcal{O}(n^3 \log n)$-time algorithm [15]. Finally, Amir, Landau, Marcus, and Sokol [10] introduced the notion of maximal repetitions in 2D strings, as an analogue of so-called runs in 1D strings.

Two tandems $T = W^{1,2}$ and $T' = V^{1,2}$ are distinct when $W \neq V$. Similarly, two quartics $Q = W^{2,2}$ and $Q' = V^{2,2}$ are distinct when $W \neq V$. It is easy to see that an $n \times n$ 2D string contains $\mathcal{O}(n^3)$ distinct tandems by applying the bound on the number of 1D distinct squares on every horizontal slice of the 2D string. It is also not hard to show that this bound is asymptotically tight, even over a binary alphabet [49]. Thus, tandems do not seem to be the right generalization of squares, and we should rather focus on quartics.

Recently, Charalampopoulos, Radoszewski, Rytter, Waleń, and Zuba [27] showed a non-trivial upper bound of $\mathcal{O}(n^2 \log^2 n)$ on the number of distinct quartics in an $n \times n$ 2D string, and an algorithm that finds them in the same time complexity. At this point, it was quite unclear to what extent distinct quartics suffer from the "curse of dimensionality". Could it be that, similarly to the number of distinct squares, their number is also linear in the size of the input? Gawrychowski, Ghazawi, and Landau [49] very recently showed that this is not the case, by constructing an infinite family of $n \times n$ 2D strings over a binary alphabet containing $\Omega(n^2 \log n)$ distinct quartics. This shows that there is a qualitative difference between distinct squares and distinct quartics, but leaves a significant gap between the lower bound of $\Omega(n^2 \log n)$ and the upper bound of $\mathcal{O}(n^2 \log^2 n)$ [27].

**Our Results.** Our contribution is twofold. First, we show an asymptotically tight bound of $\mathcal{O}(n^2 \log n)$ on the number of distinct quartics in an $n \times n$ 2D string. Thus, the "curse of dimensionality" for this problem is a single logarithm for going from 1D to 2D. Second, we show how to find all distinct quartics in worst-case optimal $\mathcal{O}(n^2 \log n)$ time. We thus resolve both the combinatorial and algorithmic complexity of distinct quartics.

A notable difference of our algorithm from the previously fastest algorithm for computing

distinct quartics [27] is that the algorithm of [27] first finds all 2D runs[1] of the 2D string, which are not even known to be $\mathcal{O}(n^2 \log n)$, and then infers the quartics from those. We manage to circumvent this, by focusing on some selected occurrences of 2D strings of the form $Q^{5,5}$, instead of considering all of them via 2D runs.

**Overview of the Combinatorial Upper Bound.** When bounding the number of distinct squares, one begins with fixing the rightmost occurrence of every distinct square [47]. In two dimensions, it is less clear what an extreme occurrence could mean. We simply say that it is an occurrence at a position $(i, j)$ such that there is no other occurrence at a different position $(i', j')$ such that $i' \geq i$ and $j' \geq j$. Next, a standard trick used when working with strings is to partition them into groups with length in $[2^a \,..\, 2^{a+1})$ for different integers $a$. Similarly to previous work [27], we partition quartics into groups $C_{(a,b)}$ with height in $[2^a \,..\, 2^{a+1})$ and width in $[2^b \,..\, 2^{b+1})$ for pairs of integers $(a, b)$. We begin with proving that, for any position $(i, j)$, the set of extreme occurrences at $(i, j)$ may have a non-empty intersection with only $\mathcal{O}(\log n)$ such groups. Next, we partition all quartics into *thin* and *thick* (note that the meaning of thin and thick is slightly different than in the previous work [27]). More specifically, a quartic $Q$ is thick if and only if it can be partitioned into $x \times y$ occurrences of a primitive 2D string $R$, i.e., $Q = R^{x,y}$ for some $x, y \geq 5$. Then, we show that for any position $(i, j)$ and group $C_{(a,b)}$, there can be at most 10 extreme occurrences of thin quartics in $C_{(a,b)}$ at position $(i, j)$. Overall, we thus have only $\mathcal{O}(n^2 \log n)$ distinct thin quartics.

The main part of our proof for the combinatorial upper bound is the analysis of the number of distinct thick quartics. Our starting point is the observation (already present in [27]) that this number can be upper bounded by the number of occurrences of 2D strings of the form $R^{5,5}$, for primitive $R$, that participate in the partition of an extreme occurrence of some quartic $R^{x,y}$. To bound the number of such occurrences, we assign an occurrence of $R^{5,5}$ at position $(i, j)$ to position $(x, y) = (i + 2 \cdot \mathsf{height}(R), j + 2 \cdot \mathsf{width}(R))$ and say that this occurrence is *anchored* at position $(x, y)$. Then, our goal is to show that the number of occurrences assigned to every position is only $\mathcal{O}(\log n)$. For a fixed position $(i, j)$, this is done by first arguing that the pairs $(\lfloor \log(\mathsf{height}(R)) \rfloor, \lfloor \log(\mathsf{width}(R)) \rfloor)$ are pairwise distinct among occurrences of different $R^{5,5}$ assigned to $(i, j)$. This requires a careful analysis of the implied periodic structure and allows us to focus on bounding the number of such pairs. What we do next is the main novelty of our approach for the combinatorial upper bound. We treat the pairs as a set of points $\mathcal{P} \subseteq [1 \,..\, m]^2$, where $m = \lfloor \log n \rfloor$, and argue that, for each $(a, b) \in \mathcal{P}$, the set of points of $\mathcal{P}$ that are strictly dominated by $(a, b)$ can be partitioned into at most two chains. Next, our goal is to upper bound the size of any set $\mathcal{P}$ with this property by $\mathcal{O}(m)$. To this end, we leverage a result from extremal combinatorics, namely, the proof of the Füredi-Hajnal conjecture by Marcus and Tardos [69]. This result states that, if an $m \times m$ binary matrix $M$ avoids a fixed permutation matrix $P$ as a submatrix, i.e., if $P$ cannot be obtained by deleting some rows and columns of $M$ and changing some 1s to 0s, then $M$ contains at most $c_P \cdot m$ 1s, where $c_P$ is a constant if the size of $P$ is a constant. We reformulate the constraint on $\mathcal{P}$ to avoid the permutation matrix shown below as a submatrix. Overall, this allows us to conclude that the number of extreme occurrences of thick quartics is also $\mathcal{O}(n^2 \log n)$.

[1] 2D runs are subarrays that are periodic both vertically and horizontally and cannot be extended without any of the periods changing.

| | | | |
|---|---|---|---|
| | | 1 | |
| | 1 | | |
| 1 | | | |
| | | | 1 |

A high-level description of our approach for the algorithmic part is provided in Section 4, as it is best read after the full proof of the combinatorial upper bound.

**Open Problem.** An interesting follow-up question on repetitions in 2D strings is that of settling the number of 2D runs that a 2D string can have. Charalampopoulos et al. [27] proved an $\mathcal{O}(n^2 \log^2 n)$ upper bound for the number of 2D runs that an $n \times n$ 2D string can contain, while Gawrychowski et al. [49] constructed an infinite family of $n \times n$ 2D strings (over a binary alphabet) with $\Omega(n^2 \log n)$ 2D runs. On the algorithms' side, Amir et al. [10] devised an algorithm that computes all 2D runs in an $n \times n$ 2D string in $\mathcal{O}(n^2 \log n + |\mathsf{output}|)$ time, and is thus optimal. For 1D strings, after a long line of results [35, 36, 52, 53, 63, 71, 74] the number of runs was shown to be less than $n$ [19] and they can be computed in $\mathcal{O}(n)$ time for strings over ordered alphabets [44] (see [19, 63] for earlier algorithms for strings over linear-time sortable alphabets).

## 2 Preliminaries

For integers $i, j \in \mathbb{Z}$, we denote the set $\{k \in \mathbb{Z} : i \leq k \leq j\}$ by either of $[i\,..\,j]$, $(i-1\,..\,j+1)$, $[i\,..\,j+1)$, and $(i-1\,..\,j]$.

Let us consider a string $S = S[1]S[2]\cdots S[n]$ of length $|S| = n$. For integers $i \leq j$ in $[1\,..\,n]$, we denote the *fragment* $S[i]\cdots S[j]$ by $S[i\,..\,j]$. A positive integer $p \leq n$ is a *period* of $S$ if and only if $S[i] = S[i+p]$ for all $i \in [1\,..\,n-p]$. The smallest period of $S$ is called *the period* of $S$ and is denoted by $\mathsf{per}(S)$. A string is called *periodic* if and only if its period is at most half its length. We will extensively use the following property of periods.

▶ **Lemma 2.1** (Periodicity Lemma [46]). *If $p$ and $q$ are periods of a string $S$ and satisfy $p + q \leq |S|$, then $\gcd(p, q)$ is also a period of $S$.*

We denote the concatenation of two strings $U$ and $V$ by $UV$. Further, for $k \in \mathbb{Z}_+$, we denote the concatenation of $k$ copies of $U$ by $U^k$. A string $V$ that cannot be written as $U^k$ for a string $U$ and an integer $k > 1$ is called *primitive*. A string of the form $UU$ is called a *square*. A square $UU$ is said to be primitively rooted if $U$ is primitive. More generally, a string of the form $U^k$ is called a $k$-th power, and it is said to be primitively rooted if $U$ is primitive. We extensively use the following property of squares.

▶ **Lemma 2.2** (Three Squares Lemma [41][2]). *If squares $U^2$ and $V^2$ are proper prefixes of a square $W^2$, $|U| < |V|$, and $U$ is primitive, then $|U| + |V| \leq |W|$.*

We next summarise some combinatorial properties of squares and higher powers.

▶ **Proposition 2.3.** *Consider a string $S$ and an integer $a$. At most two prefixes of $S$ with lengths from $[2^a\,..\,2^{a+1})$ can be primitively rooted squares.*

**Proof.** Assume that there are three such prefixes, and denote them by $UU$, $VV$, $WW$, where $|U| < |V| < |W|$. Since $|UU|, |VV|, |WW| \in [2^a\,..\,2^{a+1})$, we have $|U| + |V| > |W|$, which together with the primitivity of $|U|$ leads to a contradiction. ◀

[2] This formulation comes from [47].

| | | | | |
|---|---|---|---|---|
| W | W | W | W | W |
| W | W | W | W | W |

**Figure 1** 2D string $W^{2,5}$ is shown for some 2D string $W$.

▶ **Proposition 2.4.** *Consider a string $S$ and an integer $a$. All prefixes of $S$ with lengths in $[2^a\,..\,2^{a+1})$ that are powers higher than 2 are of the form $U^k$ for the same primitive string $U$.*

**Proof.** Assume that there are two such prefixes $U^k$ and $V^\ell$, where $U$ is primitive, $k, \ell \geq 3$, and $|U| < |V|$. First, $|V|$ is a period of $Z := S[1\,..\,|U| + |V|]$. Second, $|V| < 2^{a+1}/3$ and consequently $|U^{k-1}| \geq |V|$, as otherwise we would have $|U^k| < k \cdot |V|/(k-1) \leq 3|V|/2 < 2^a$, hence $|U|$ is also a period of $Z$. We thus have that both $|U|$ and $|V|$ are periods of $Z$. Then, an application of Lemma 2.1 yields that $\gcd(|U|, |V|)$ is a period of $Z$. But $U$ is primitive so $\gcd(|U|, |V|) = |U|$, and $V$ hence is a power of $U$, a contradiction. ◀

A fragment $S[i\,..\,j]$ of a string $S$ is a *run* if and only if it is periodic and it cannot be extended by a character in either direction with its period remaining unchanged.

An $m \times n$ 2D string $A$ is simply a two-dimensional array with $m$ rows and $n$ columns, where $\mathsf{height}(A) = m$ and $\mathsf{width}(A) = n$. The position that lies on the $i$-th row and the $j$-th column of $A$ is position $(i, j)$. We regard the top-left position of a 2D string as position $(1, 1)$, and the bottom-right position as position $(m, n)$. That is, we index rows from top to bottom and columns from left to right. We say that a 2D string $P$ occurs at a position $(i, j)$ of a 2D string $T$ if and only if the *subarray* (also called *fragment*) $T[i\,..\,i+\mathsf{height}(P))[j\,..\,j+\mathsf{width}(P))$ of $T$ equals $P$. We write $\Sigma^{*,*}$ to denote the set of all 2D strings over alphabet $\Sigma$.

A positive integer $p$ is a *horizontal period* of a 2D string $A$ such that $\mathsf{width}(A) \geq p$ if and only if the $j$-th column of $A$ is equal to the $(j+p)$-th column of $A$ for all $j \in [1\,..\,\mathsf{width}(A)-p]$. The smallest horizontal period of $A$ is *the horizontal period* of $A$. An integer $q$ is a (the) *vertical period* of $A$ if and only if $q$ is a (resp. the) horizontal period of the transpose of $A$.

It will be sometimes convenient to view a 2D string as a 1D metastring by viewing each column (or row) as a metacharacter such that metacharacters are equal if and only if the corresponding columns (resp. rows) are equal. Observe that the horizontal periods (resp. vertical periods) of a 2D string $A$ are in one-to-one correspondence with the periods of the metastring obtained from $A$ by viewing each column (resp. row) as a metacharacter.

For a 2D string $W$ and $x, y \in \mathbb{Z}_+$, we denote by $W^{x,y}$ the 2D string that consists of $x \times y$ copies of $W$; see Figure 1 for an illustration. A 2D string $W$ is *primitive* if it cannot be written as $Y^{a,b}$ for any 2D string $Y$ and $a, b \in \mathbb{Z}_+$ that are not both equal to 1. The primitive root of a 2D string $X$ is the unique primitive 2D string $Y$ such that $X = Y^{a,b}$ for $a, b \in \mathbb{Z}_+$. Note that the primitive root is indeed unique by the periodicity lemma applied to the horizontal and vertical 1D metastrings obtained from $X$.

**Model of computation.** For our algorithm, we assume the standard word-RAM model of computation with word-size $\Omega(\log n)$, where $n$ is the size of the input.

## 3 The Combinatorial Bound

We consider an $n \times n$ 2D string $A$, whose entries are over an arbitrary alphabet $\Sigma$. We say that a fragment $A[i\,..\,i')[j\,..\,j')$ is a quartic-fragment if and only if it equals some quartic $Q$; further, we say that it is an *extreme* or *bottom-right* quartic-fragment if $Q$ does not have any occurrence at another position $(i'', j'')$ with $i'' \geq i$ and $j'' \geq j$. We refer to such an

occurrence of $Q$ as an extreme or bottom-right occurrence. We denote by $\mathsf{BR}(i,j)$ the set of extreme quartic-fragments with top-left corner $(i,j)$. Further, we denote the union of all $\mathsf{BR}(i,j)$ by $\mathsf{BR}$. Observe that the distinct quartics in $\mathsf{BR}$ are exactly the distinct quartics in $A$ as every quartic that occurs in $A$ has at least one extreme occurrence. Note that a quartic may have $\Theta(n)$ extreme occurrences; an example is provided in Figure 2.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 2** Consider an $n \times n$ 2D string $A$ all of whose entries that lie weakly above the main diagonal are equal to 0 and all of whose entries that lie strictly below the main diagonal are equal to 1. The quartic that equals $0^{2,2}$ has $n-2$ extreme occurrences in $A$. This is illustrated for $n=8$: the bottom-right corners of extreme occurrences of said quartic are marked.

Let us consider a partition of the quartic-fragments of $A$ into $\mathcal{O}(\log^2 n)$ *canonical sets*, such that, for each $(a,b) \in [1\,..\,\lfloor \log n \rfloor]^2$, the canonical set $C_{(a,b)}$ consists of all quartic-fragments of $A$ whose height is in $[2^a\,..\,2^{a+1})$ and whose width is in $[2^b\,..\,2^{b+1})$.[3]

▶ **Lemma 3.1.** *For each position $(i,j)$ of $A$, $\mathsf{BR}(i,j)$ has a non-empty intersection with $\mathcal{O}(\log n)$ canonical sets.*

**Proof.** We say that the *aspect ratio* of a quartic $Q$ is equal to 2 raised to the power $\lfloor \log(\mathsf{height}(Q)) \rfloor - \lfloor \log(\mathsf{width}(Q)) \rfloor$. The aspect ratio of all quartic-fragments in a canonical set $C_{(a,b)}$ is $2^{a-b}$. Observe, that there are $2 \cdot \lfloor \log n \rfloor - 1$ different possible values for the aspect ratio of a quartic. For each $d \in [-\lfloor \log n \rfloor + 1\,..\,\lfloor \log n \rfloor - 1]$, let $\mathsf{BR}_d(i,j)$ be the subset of $\mathsf{BR}(i,j)$ that contains exactly the elements of $\mathsf{BR}(i,j)$ with aspect ratio $2^d$.

Next, we show that, for each $d$, we have at most two canonical sets contributing to $\mathsf{BR}_d(i,j)$. Let us suppose towards a contradiction that we have three canonical sets $C_{(a,b)}$, $C_{(a',b')}$, and $C_{(a'',b'')}$ that contribute to $\mathsf{BR}_d(i,j)$. In other words, there are quartic-fragments

- $Q \in \mathsf{BR}_d(i,j)$ with height in $[2^a\,..\,2^{a+1})$ and width in $[2^b\,..\,2^{b+1})$,
- $Q' \in \mathsf{BR}_d(i,j)$ with height in $[2^{a'}\,..\,2^{a'+1})$ and width in $[2^{b'}\,..\,2^{b'+1})$, and
- $Q'' \in \mathsf{BR}_d(i,j)$ with height in $[2^{a''}\,..\,2^{a''+1})$ and width in $[2^{b''}\,..\,2^{b''+1})$.

Since $a-b = a'-b' = a''-b'' = d$, we can assume without loss of generality that $a < a' < a''$ and $b < b' < b''$. We thus have that $a+1 < a''$ and $b+1 < b''$, which implies that $Q$ is fully contained in the top left quarter of $Q''$. Thus, $Q$ has an occurrence at position $(i + \mathsf{height}(Q'')/2, j + \mathsf{width}(Q'')/2)$; see Figure 3. This contradicts our assumption that the occurrence of $Q$ at position $(i,j)$ is an extreme occurrence.

Thus, $\mathcal{O}(\log n)$ canonical sets contribute to $\mathsf{BR}(i,j)$: at most two for each aspect ratio. ◀

Henceforth, we call a quartic $Q$ with primitive root $P$ *thick* if $Q = P^{x,y}$ for $x, y \geq 5$ and *thin* otherwise.

[3] Throughout this work, logarithms have base 2.

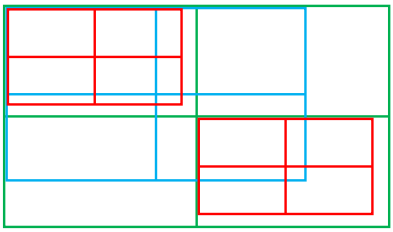

**Figure 3** An illustration of the proof of Lemma 3.1 with quartics $Q$, $Q'$, and $Q''$ drawn in red, blue, and green, respectively.

▶ **Lemma 3.2.** *For any position $(i, j)$ of $A$ and any pair $(a, b) \in [1 \,..\, \lfloor \log n \rfloor]^2$, $C_{(a,b)} \cap \mathsf{BR}(i, j)$ can contain at most 10 thin quartics.*

**Proof.** The possible forms of *thin* quartics are $P^{2,2}$, $P^{2,2x}$, $P^{2y,2}$, $P^{4,2x}$, and $P^{2y,4}$ for $x > 1$ and $y > 1$. We will consider each form separately.

First, we consider quartics of the form $P^{2,2}$ in $C_{(a,b)} \cap \mathsf{BR}(i, j)$. We analyse the fragment $A[i \,..\, n][j \,..\, j + 2^b)$ and treat it as a metastring by viewing rows as metacharacters. We observe that each considered quartic defines a prefix of this string that is a square. Further, all those squares need to be primitive, as otherwise $P$ could be written as $P = Q^{k,1}$, for some $k > 1$, in contradiction with the primitivity of $P$. Thus, by Proposition 2.3 we have at most two possible heights for the considered quartics. By a symmetric argument, we have at most two possible widths, and so at most 4 quartics.

Second, we consider quartics of the form $P^{2,2x}$ in $C_{(a,b)} \cap \mathsf{BR}(i, j)$. By the same reasoning as above, we have at most two possible heights for the considered quartics; let $h$ by one of them. We analyse the fragment $A[i \,..\, i + h)[j \,..\, n]$ and treat it as a metastring by viewing columns as metacharacters. Each considered quartic with height $h$ corresponds to a prefix that is a $(2x)$-th power, for some $x > 1$. By Proposition 2.4, all such prefixes are powers of the same $U$; let $U^{2x}$ be the longest such prefix. Then, for any $x' < x$, the prefix $U^{2x'}$ also occurs at position $(i, j + |U|)$, so the occurrence at position $(i, j)$ cannot be an extreme occurrence. Therefore, for every possible height, we have at most one quartic, so at most 2 in total.

Third, we consider quartics of the form $P^{4,2x}$ for $x > 1$ in $C_{(a,b)} \cap \mathsf{BR}(i, j)$. We (again) analyse the fragment $A[i \,..\, n][j \,..\, j + 2^b)$ and treat it as a metastring by viewing its rows as metacharacters. We observe that each considered quartic defines a prefix that is a primitively rooted fourth power there. Thus, by Proposition 2.4 we have at most one possible height, and, by the same reasoning, as above at most one quartic.

Symmetric arguments bound the number of quartics of the forms $P^{2y,2}$ and $P^{2y,4}$. ◀

▶ **Lemma 3.3.** *The number of distinct thin quartics in $A$ is $\mathcal{O}(n^2 \log n)$.*

**Proof.** For each position $(i, j)$ of $A$, $\mathsf{BR}(i, j)$ has a non-empty intersection with at most $\mathcal{O}(\log n)$ canonical sets due to Lemma 3.1. Further, by Lemma 3.2, there are at most 10 thin quartics in each such intersection. Since $A$ has $n^2$ positions, the stated bound follows. ◀

## 3.1 Reduction to a Geometric Problem

We next partition the thick quartics by primitive root. For each primitive 2D string $R$, we choose any bottom-right occurrence of each distinct thick quartic with primitive root $R$. We denote the obtained set of quartic-fragments by $\mathsf{Thick}_R$. Additionally, let us denote

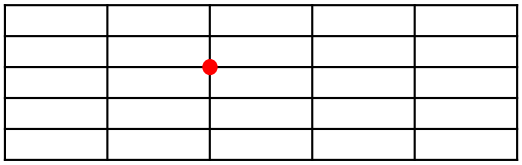

**Figure 4** The red point corresponds to the anchor of the shown occurrence of $R^{5,5}$.

by $\mathsf{occ}_{5\times5}(R)$ the set of all positions $(i,j)$ of $A$ where $R^{5,5}$ occurs such that there is an element of $\mathsf{Thick}_R$ that fully contains this occurrence of $R^{5,5}$ and has top-left corner equal to $(i - x\cdot\mathsf{height}(R), j - y\cdot\mathsf{width}(R))$ for some non-negative integers $x$ and $y$.

The proof of the following lemma proceeds almost exactly as the proof of Claim 18 in [27], except that we work with occurrences of $R^{5,5}$ instead of $R^{3,3}$ and do not need the notion of special points. We provide a detailed description for completeness.

▶ **Lemma 3.4** (cf. the proof of [27, Claim 18]). *For any 2D string $R$, $|\mathit{Thick}_R| \le |\mathit{occ}_{5\times5}(R)|$.*

**Proof.** We will map each $Q \in \mathsf{Thick}_R$ to an occurrence of $R^{5,5}$ in such a way that two distinct quartic-fragments $Q, Q' \in \mathsf{Thick}_R$ are mapped to distinct occurrences. This will imply that the number of occurrences of $R$ is at least as large as the number of elements of $\mathsf{Thick}_R$.

For each $x = 6, 8, \ldots$ in this order, we select $Q \in \mathsf{Thick}_R$ such that $\mathsf{height}(Q) = x\cdot\mathsf{height}(R)$ and $\mathsf{width}(Q) = y\cdot\mathsf{width}(R)$ is the largest among all $Q' \in \mathsf{Thick}_R$ with $\mathsf{height}(Q') = x\cdot\mathsf{height}(R)$. We note that the number of $Q' \in \mathsf{Thick}_R$ with $\mathsf{height}(Q') = x\cdot\mathsf{height}(R)$ is at most $y/2-2$, and our goal is to map them to occurrences of $R^{5,5}$ that have not been used so far. Additionally, we will ensure that those occurrences are all in the same row. Let $(i,j)$ be the position of an extreme occurrence of $Q$. We observe that $R^{5,5}$ occurs at every position $(i',j')$ with $i' = i + k\cdot\mathsf{height}(R)$ and $j' = j + \ell\cdot\mathsf{width}(R)$, for every $k \in [0\,..\,x-5]$ and $\ell \in [0\,..\,y-5]$. We choose $k \in [0\,..\,x-5]$ such that none of the occurrences of $R^{5,5}$ at positions $(i + k\cdot\mathsf{height}(R), j + \ell\cdot\mathsf{width}(R))$, for $\ell = 0, 1, \ldots, y-3$ have been used so far. This is possible because so far we have used occurrences of $R^{5,5}$ in only $x/2-3 < x-4$ rows. Then, we map every $Q' \in \mathsf{Thick}_R$ with $\mathsf{height}(Q') = x\cdot\mathsf{height}(R)$ to an occurrence of $R^{5,5}$ at position $(i + k\cdot\mathsf{height}(R), j + \ell\cdot\mathsf{width}(R))$, for some $\ell \in [0\,..\,y-5]$, which is possible due to $y/2-2 \le y-4$. ◀

Thus, it remains to upper bound $\sum_R |\mathsf{occ}_{5\times5}(R)|$, i.e., the sum, over all $R$, of the number of occurrences of $R^{5,5}$ which are contained in some element of $\mathsf{Thick}_R$.

Consider an occurrence of a 2D string of the form $R^{5,5}$, for a primitive string $R$, at a position $(i,j)$ of $A$. We call position $(i + 2\cdot\mathsf{height}(R), j + 2\cdot\mathsf{width}(R))$ the *anchor* of this occurrence; see Figure 4.

Now, for each primitive string $R$, for each element of $\mathsf{occ}_{5\times5}(R)$, we assign the corresponding occurrence of $R^{5,5}$ to its anchor. Let $\mathsf{assign}(i,j)$ be the set of primitive 2D strings $R$ such that an occurrence of $R^{5,5}$ has been assigned to position $(i,j)$. We have

$$\sum_R |\mathsf{occ}_{5\times5}(R)| = \sum_{i=1}^{n}\sum_{j=1}^{n} |\mathsf{assign}(i,j)|. \tag{1}$$

It now suffices to show that $\sum_{i=1}^{n}\sum_{j=1}^{n} |\mathsf{assign}(i,j)| = \mathcal{O}(n^2\log n)$. We will show that $|\mathsf{assign}(i,j)| = \mathcal{O}(\log n)$ for all $i,j$, which straightforwardly yields the desired bound.

Let us fix a position $(i,j)$. By applying Proposition 2.4 horizontally and vertically one easily obtains the following fact.

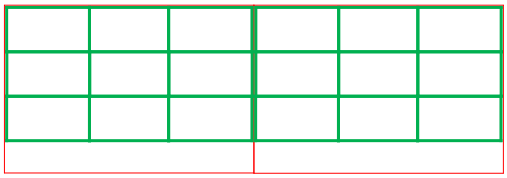

**Figure 5** The red rectangles correspond to $T^{1,2}$ and the green rectangles correspond to $S^{3,6}$.

▶ **Fact 3.5** ([27, Corollary 13]). *Let $a$, $b$ be non-negative integers and $W, Z$ be different 2D strings with height in $[2^a\,..\,2^{a+1})$ and width in $[2^b\,..\,2^{b+1})$. If $W^{3,3}$ and $Z^{3,3}$ occur at some position of $A$, then at least one of $W$ and $Z$ is not primitive.*

This, together with the fact that, for each $R \in \mathsf{assign}(i,j)$, $R^{3,3}$ occurs at position $(i,j)$, implies the following.

▶ **Fact 3.6.** *For each pair $(a,b) \in [1\,..\,\lfloor \log n \rfloor]^2$, for each position $(i,j)$ of $A$, the set $\mathsf{assign}(i,j)$ contains at most one element with height in $[2^a\,..\,2^{a+1})$ and width in $[2^b\,..\,2^{b+1})$.*

Let us define a map $\Sigma^{*,*} \to [1\,..\,\lfloor \log n \rfloor]^2$ as $g(R) \mapsto (\lfloor \log(\mathsf{height}(R)) \rfloor, \lfloor \log(\mathsf{width}(R)) \rfloor)$. Let $f$ be the restriction of $g$ to the domain $\mathsf{assign}(i,j)$. Due to Fact 3.6, $f$ is an injective function. We henceforth identify each element $R$ of $\mathsf{assign}(i,j)$ with point $f(R)$. We denote the image of $f$ by $\mathcal{P}$.

We say that a point $(a,b) \in \mathbb{Z}^2$ *dominates* or *weakly dominates* a point $(a',b')$ if $a' \le a$ and $b' \le b$; the dominance is *strict* if $a' < a$ and $b' < b$. (If we require some dominance to be strict we explicitly say so; that is, whenever we refer to some dominance without explicitly mentioning whether it is weak or strict, we refer to weak dominance.) A set of points on which the domination relation forms a total order is called a *chain*. A set of points such that none dominates another is called an *antichain*. We are going to use Dilworth's theorem [43], which states that, in any finite partially ordered set, the size of the largest antichain is equal to the minimum number of chains in which the elements of the set can be decomposed.

For two primitive 2D strings $S$ and $T$, with $3 \cdot \mathsf{height}(S) < \mathsf{height}(T)$ and $\mathsf{width}(S) < \mathsf{width}(T)$, we say that $S$ *horizontally spans* $T$ when the 2D string $\mathsf{row}_{3\cdot\mathsf{height}(S)}(T^{1,2})$, consisting of the $3 \cdot \mathsf{height}(S)$ topmost rows of $T^{1,2}$, equals $S^{3,y}$ for some even integer $y \ge 4$; see Figure 5. Similarly, when $\mathsf{width}(S) < \mathsf{width}(T)$ and $\mathsf{height}(S) < \mathsf{height}(T)$, we say that $S$ *vertically spans* $T$ when the 2D string $\mathsf{col}_{3\cdot\mathsf{width}(S)}(T^{2,1})$, consisting of the $3 \cdot \mathsf{width}(S)$ leftmost columns of $T^{2,1}$, equals $S^{x,3}$ for some even integer $x \ge 4$.

▶ **Fact 3.7.** *Let $S$ and $T$ be two primitive 2D strings. If $S$ spans $T$ horizontally, then the horizontal period of $\mathsf{row}_{3\cdot\mathsf{height}(S)}(T^{1,2})$ is $\mathsf{width}(S)$. Symmetrically, if $S$ spans $T$ vertically, then the vertical period of $\mathsf{col}_{3\cdot\mathsf{width}(S)}(T^{2,1})$ is $\mathsf{height}(S)$.*

**Proof.** We only prove the first statement as the second one follows by symmetry. Let us view $\mathsf{row}_{3\cdot\mathsf{height}(S)}(T^{1,2})$ as a metastring $Z$ by viewing each of its columns as a metacharacter; the horizontal period of $\mathsf{row}_{3\cdot\mathsf{height}(S)}(T^{1,2})$ equals $p := \mathsf{per}(Z)$. Note that $\mathsf{width}(S)$ is a period of $Z$. Towards a contradiction, suppose that $p < \mathsf{width}(S)$. Then, an application of the periodicity lemma to $Z$ implies that $p$ must divide $\mathsf{width}(S)$. This fact contradicts the primitivity of $S$, as we would have that $S = (S[1\,..\,\mathsf{height}(S)][1\,..\,p])^{1,k}$ for $k = \mathsf{width}(S)/p$; see Figure 5. ◀

When reading the following lemma, one can think of $M$ being in $\mathsf{assign}(i,j)$. However, the lemma is slightly more general, as needed for the algorithm that is presented in the full version.

▶ **Lemma 3.8.** *Let $R \in \mathsf{assign}(i,j)$ and $M$ be a primitive 2D string such that:*
- *$M^{5,5}$ has an occurrence with anchor $(i,j)$;*
- *$g(M)$ strictly dominates $g(R)$.*

*Then, $R$ spans $M$ either horizontally or vertically (or both).*

**Proof.** By the definition of $\mathsf{assign}(i,j)$, the occurrence of $R^{5,5}$ assigned to position $(i,j)$ appears inside an element of $\mathsf{Thick}_R$, which is necessarily a bottom-right occurrence of a thick quartic with primitive root $R$. Let us denote this quartic by $Q$. Observe that the considered occurrence of $Q$ cannot be fully contained inside the occurrence of $M^{4,4}$ at position $(i-2\cdot\mathsf{height}(M), j-2\cdot\mathsf{width}(M))$ as this would contradict the fact that the considered occurrence of $Q$ is bottom-right: there would be another occurrence $\mathsf{width}(M)$ positions to the right. Therefore, $Q$ must contain at least one of the following four fragments of $A$, depicted in Figure 6: $A[i\,..\,i{+}3{\cdot}\mathsf{height}(R))[j\,..\,j{+}2{\cdot}\mathsf{width}(M))$, $A[i\,..\,i{+}3{\cdot}\mathsf{height}(R))[j{-}2{\cdot}\mathsf{width}(M)\,..\,j)$, $A[i\,..\,i+2\cdot\mathsf{height}(M))[j\,..\,j+3\cdot\mathsf{width}(R))$, $A[i-2\cdot\mathsf{height}(M)\,..\,i)[j\,..\,j+3\cdot\mathsf{width}(R))$.

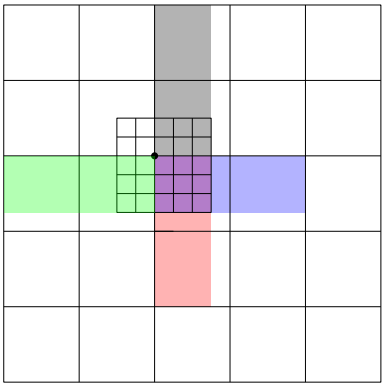

**Figure 6** The considered occurrences of each of $M^{5,5}$ and $R^{5,5}$ are shown, together with the four specified fragments, at least one of which must be fully contained in $Q$.

We next show that in either of the first two cases, $R$ horizontally spans $M$. In the remaining cases, a symmetric argument yields that $R$ vertically spans $M$. First, observe that we have $\mathsf{width}(R) < \mathsf{width}(M)$ as a direct consequence of $g(M)$ strictly dominating $g(R)$. We next need to argue that $3\mathsf{height}(R) < \mathsf{height}(M)$. If this were not the case, $\mathsf{width}(R)$ would be a horizontal period of $M^{1,2}$, contradicting the primitivity of $M$. This completes the proof. ◀

▶ **Lemma 3.9.** *Two primitive 2D strings $R_1$ and $R_2$ such that $g(R_1)$ and $g(R_2)$ form an antichain cannot both horizontally span a 2D string $R$.*

**Proof.** Let $g(R_1) = (a_1, b_1)$ and $g(R_2) = (a_2, b_2)$. Without loss of generality, we can assume that $a_1 < a_2$ and $b_1 > b_2$. Suppose towards a contradiction that both $R_1$ and $R_2$ horizontally span $R$. By applying Fact 3.7, we obtain that
- $\mathsf{width}(R_1)$ is the horizontal period of $\mathsf{row}_{3\cdot\mathsf{height}(R_1)}(R^{1,2})$ and
- $\mathsf{width}(R_2)$ is the horizontal period of $\mathsf{row}_{3\cdot\mathsf{height}(R_2)}(R^{1,2})$.

Note that, for any $k \in [1\,..\,\mathsf{height}(R)]$, the horizontal period of the string comprised of the $k$ topmost rows of $R^{1,2}$ equals the least common multiple of the periods of those $k$ rows. Hence, the period cannot decrease as we increase the number of considered rows. We thus have $\mathsf{width}(R_1) \le \mathsf{width}(R_2)$ since our assumption that $a_2 > a_1$ implies that $\mathsf{height}(R_2) > \mathsf{height}(R_1)$. This is a contradiction to our assumption that $b_1 > b_2$, which implies that $\mathsf{width}(R_1) > \mathsf{width}(R_2)$. ◀

The above lemma, Fact 3.6, and Dilworth's theorem together imply the following.

▶ **Corollary 3.10.** *All primitive 2D strings that span a primitive 2D string $R$ can be decomposed to two sets $H$ and $V$, such that*
- *the elements of $H$ span $R$ horizontally;*
- *the elements of $V$ span $R$ vertically;*
- *the restriction of $g$ to $H \cup V$ is an injective function;*
- *each of the sets $g(H)$ and $g(V)$ is a chain.*

Finally, as mentioned in the introduction, we need the following purely geometric lemma that follows from the result of Marcus and Tardos [69] on the number of 1s in an $m \times m$ binary matrix $M$ that avoids a fixed permutation $P$ as a submatrix.

▶ **Lemma 3.11.** *Consider a positive integer $m$ and a set $\mathcal{P} \subseteq [1\,..\,m]^2$. If, for each $p \in \mathcal{P}$, the set of points of $\mathcal{P}$ that are strictly dominated by $p$ can be partitioned into at most two chains, then $|\mathcal{P}| = \mathcal{O}(m)$.*

**Proof.** We think of $\mathcal{P}$ as an $m \times m$ matrix $M[1\,..\,m][1\,..\,m]$, where $M[a][b] = 1$ when $(a, b) \in \mathcal{P}$ and $M[a, b] = 0$ otherwise. Next, we say that $M$ contains a matrix $P$ as a submatrix when $P$ can be obtained from $M$ by removing rows, removing columns, and changing 1s into 0s. We claim that, by the assumptions in the lemma, $M$ does not contain the following matrix $P$ as a submatrix:

| | | | |
|---|---|---|---|
| | | 1 | |
| | 1 | | |
| 1 | | | |
| | | | 1 |

To establish this, assume otherwise towards a contradiction. Then, there exists $(a, b) \in \mathcal{P}$ and $(a_1, b_1), (a_2, b_2), (a_3, b_3) \in \mathcal{P}$ such that $(a, b)$ strictly dominates $(a_1, b_1), (a_2, b_2), (a_3, b_3)$ and further $(a_1, b_1), (a_2, b_2), (a_3, b_3)$ create an antichain. By Dilworth's theorem, this implies that the points in $\mathcal{P}$ dominated by $(a, b)$ cannot be partitioned into two chains, a contradiction. Thus, $M$ indeed does not contain $P$ as a submatrix. Because $P$ is a permutation matrix, this implies $|\mathcal{P}| = \mathcal{O}(m)$. ◀

We now complete the proof of our main result with the aid of Lemma 3.11.

▶ **Theorem 3.12.** *An $n \times n$ 2D string has $\mathcal{O}(n^2 \log n)$ distinct quartics.*

**Proof.** The number of distinct thin quartics is $\mathcal{O}(n^2 \log n)$ by Lemmas 3.2 and 3.3. The number of distinct thick quartics is

$$\sum_R |\mathsf{Thick}_R| \le \sum_R |\mathsf{occ}_{5\times 5}(R)| \qquad \text{(Lemma 3.4)}$$

$$\le \sum_{i=1}^{n} \sum_{j=1}^{n} |\mathsf{assign}(i, j)|. \qquad (1)$$

To conclude the proof, it remains to show that $|\mathsf{assign}(i, j)| = \mathcal{O}(\log n)$ for all $(i, j) \in [1\,..\,n]^2$. Let $m = \lfloor \log n \rfloor$, and recall that $\mathcal{P} \subseteq [1\,..\,m]^2$ was defined as the image of $f$, which in turn was the restriction of $g$ to the domain $\mathsf{assign}(i, j)$. By Fact 3.6, we only need to show that $|\mathcal{P}| = \mathcal{O}(m)$. By Lemma 3.8 and Corollary 3.10, for each $p \in \mathcal{P}$, the set of all points of $\mathcal{P}$ that are strictly dominated by $p$ can be partitioned into at most two chains. Thus, by Lemma 3.11 we conclude that indeed $|\mathcal{P}| = \mathcal{O}(m)$, concluding the proof. ◀

## 4 Overview of the Algorithm

The detailed algorithm can be found in the full version of this work. After preprocessing the input 2D string, we compute thin and thick quartics separately. Here, we provide an overview of the main ideas of our $\mathcal{O}(n^2 \log n)$-time algorithm for computing distinct quartics in an $n \times n$ 2D string $A$ over an ordered alphabet.

### Computation of Thin Quartics

For thin quartics, our algorithm is quite similar to the combinatorial analysis. For each position $(i, j)$, we compute an $\mathcal{O}(\log n)$-size superset $\mathcal{C}$ of the canonical sets that have a non-empty intersection with $\mathsf{BR}(i, j)$. We do this by relating extreme occurrences of quartics with occurrences of squares in metastrings obtained by viewing the columns of $A[i \,..\, i + 2^a)[1 \,..\, n]$, where $a \in [1 \,..\, \lfloor \log n \rfloor]$, as metacharacters. These squares can be efficiently computed and give us a handle on the sought thin quartics. Then, we compute the intersection of each canonical set in $\mathcal{C}$ with $\mathsf{BR}(i, j)$ in constant time using known tools that allow us to efficiently operate on the metastrings. We do this by fixing $(a, b) \in [1 \,..\, \lfloor \log n \rfloor]^2$ and computing all quartics with height in $[2^a \,..\, 2^{a+1})$ and width $[2^b \,..\, 2^{b+1})$ that occur at position $(i, j)$ of $A$, of the form $P^{2y,2}$, $P^{2y,4}$, $P^{2,2x}$, and $P^{4,2x}$, for a primitive 2D string $P$ and $x, y \geq 1$.

### Computation of Thick Quartics

For thick quartics, our algorithmic approach follows our combinatorial approach in a more relaxed sense. The main technical challenge is to compute, for each position $(i, j)$, an $\mathcal{O}(\log n)$-size set $\mathcal{R}$ of primitive 2D strings $R$, such that $\mathsf{assign}(i, j) \subseteq \mathcal{R}$. Then, those supersets can be postprocessed as in [27] in time linear in their total size to yield the sought distinct thick quartics. The computation of $\mathcal{R}$ is split into two major steps outlined next.

**Skyline Computation.** First, we compute a set $\mathcal{S}$ of *skyline primitive 2D strings* such that $S \in \mathcal{S}$ when (a) $S^{5,5}$ has an occurrence anchored at position $(i, j)$, and (b) there is no other primitive 2D string $T$ with $g(T) \geq g(S)$ such that $T^{5,5}$ has an occurrence anchored at position $(i, j)$. This part of the proof is quite technical: it heavily relies on the analysis of periodicity for 1D (meta)strings and, roughly speaking, on the analysis of the evolution of the horizontal periodic structure of a 2D string as rows are appended to it. We show that, for each $a \in [1 \,..\, \lfloor \log n \rfloor]$, there is a single candidate $h \in [2^a \,..\, 2^{a+1})$ to be considered as the height of an element of $\mathcal{S}$. Then, using runs in 1D metastrings whose origins in $A$ have sufficient overlap and bit-tricks, we can compute the widest 2D string $S$ with height $h$ such that an occurrence of $S^{5,5}$ is anchored at position $(i, j)$ in constant time (using batched computations), if one exists.

**Computation of Dominated 2D strings.** This turned out to be the most challenging part of our approach. For this exposition, let us treat $\Sigma^{*,*}$ as a partially ordered set, in the order of decreasing widths. Let $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ in accordance with this order. To obtain $\mathcal{R}$ from $\mathcal{S}$, we need to add to it the 2D strings $R \in \mathsf{assign}(i, j) \setminus \mathcal{S}$. By the construction of $\mathcal{S}$ we know that there exists $S \in \mathcal{S}$ such that $g(R) \leq g(S)$.

Our combinatorial analysis implies that each $R \in \mathsf{assign}(i, j)$ spans each element $S \in \mathcal{S}$ for which $g(S)$ strictly dominates $g(R)$ either vertically or horizontally. It turns out that if $R$ spans all of these elements of $\mathcal{S}$ either vertically or horizontally, it is easy to compute it efficiently. This is, unfortunately, not the case in general. However, we observe that $R$ spans

vertically (resp. horizontally) a contiguous subset of $\mathcal{S}$. We show that the problem boils down to computing the union, over all $k$, of sets $I_k$, where $I_k$ contains exactly those primitive 2D strings that span $S_{k-1}$ vertically and span $S_k$ horizontally. Crucially, we observe that due to a strong form of transitivity of the spanning property, the intersection of any two such $I_k$ and $I_{k'}$ consists of a number of the smallest elements of both (i.e., their longest common prefix if viewed as strings). Hence, by computing the elements of $I_k$ from the largest to the smallest, we can stop whenever we encounter an element that has already been reported by this procedure. This allows us to reduce the computation of $\mathcal{R}$ to the problem of efficiently computing sets $I_k$. To this end, we prove that an $\mathcal{O}(\log n)$-bits representation of the evolution of the periodic structure of certain fragments of $A$ as rows and columns are appended to them suffices for inferring $I_k$; we then use tabulation to infer it efficiently.

## References


1 Amihood Amir and Gary Benson. Efficient two-dimensional compressed matching. In *Data Compression Conference*, pages 279–288. IEEE Computer Society, 1992. doi:10.1109/DCC.1992.227453.

2 Amihood Amir and Gary Benson. Two-dimensional periodicity in rectangular arrays. *SIAM J. Comput.*, 27(1):90–106, 1998. doi:10.1137/S0097539795298321.

3 Amihood Amir, Gary Benson, and Martin Farach. An alphabet independent approach to two-dimensional pattern matching. *SIAM Journal on Computing*, 23(2):313–323, 1994. doi:10.1137/S0097539792226321.

4 Amihood Amir, Gary Benson, and Martin Farach. Optimal two-dimensional compressed matching. *J. Algorithms*, 24(2):354–379, 1997. doi:10.1006/JAGM.1997.0860.

5 Amihood Amir, Gary Benson, and Martin Farach. Optimal parallel two dimensional text searching on a CREW PRAM. *Inf. Comput.*, 144(1):1–17, 1998. doi:10.1006/INCO.1998.2705.

6 Amihood Amir, Ayelet Butman, Moshe Lewenstein, and Ely Porat. Real two dimensional scaled matching. *Algorithmica*, 53(3):314–336, 2009. doi:10.1007/S00453-007-9021-X.

7 Amihood Amir and Eran Chencinski. Faster two dimensional scaled matching. *Algorithmica*, 56(2):214–234, 2010. doi:10.1007/S00453-008-9173-3.

8 Amihood Amir and Martin Farach. Two-dimensional dictionary matching. *Inf. Process. Lett.*, 44(5):233–239, 1992. doi:10.1016/0020-0190(92)90206-B.

9 Amihood Amir and Martin Farach. Efficient 2-dimensional approximate matching of half-rectangular figures. *Inf. Comput.*, 118(1):1–11, 1995. doi:10.1006/INCO.1995.1047.

10 Amihood Amir, Gad M. Landau, Shoshana Marcus, and Dina Sokol. Two-dimensional maximal repetitions. *Theoretical Computer Science*, 812:49–61, 2020. doi:10.1016/j.tcs.2019.07.006.

11 Amihood Amir, Gad M. Landau, and Dina Sokol. Inplace 2d matching in compressed images. *J. Algorithms*, 49(2):240–261, 2003. doi:10.1016/S0196-6774(03)00088-9.

12 A. Apostolico and V.E. Brimkov. Fibonacci arrays and their two-dimensional repetitions. *Theoretical Computer Science*, 237(1-2):263–273, 2000. doi:10.1016/S0304-3975(98)00182-0.

13 Alberto Apostolico. Optimal parallel detection of squares in strings. *Algorithmica*, 8(4):285–319, 1992. doi:10.1007/BF01758848.

14 Alberto Apostolico and Dany Breslauer. An optimal $\mathcal{O}(\log \log n)$-time parallel algorithm for detecting all squares in a string. *SIAM J. Comput.*, 25(6):1318–1331, 1996. doi:10.1137/S0097539793260404.

15 Alberto Apostolico and Valentin E. Brimkov. Optimal discovery of repetitions in 2D. *Discrete Applied Mathematics*, 151(1-3):5–20, 2005. doi:10.1016/j.dam.2005.02.019.

16 Alberto Apostolico and Franco P. Preparata. Optimal off-line detection of repetitions in a string. *Theor. Comput. Sci.*, 22:297–315, 1983. doi:10.1016/0304-3975(83)90109-3.

**17** Alberto Apostolico and Franco P. Preparata. Data structures and algorithms for the string statistics problem. *Algorithmica*, 15(5):481–494, 1996.

**18** Theodore P. Baker. A technique for extending rapid exact-match string matching to arrays of more than one dimension. *SIAM Journal on Computing*, 7(4):533–541, 1978. doi:10.1137/0207043.

**19** Hideo Bannai, Tomohiro I, Shunsuke Inenaga, Yuto Nakashima, Masayuki Takeda, and Kazuya Tsuruta. The "runs" theorem. *SIAM Journal on Computing*, 46(5):1501–1514, 2017. doi:10.1137/15M1011032.

**20** Hideo Bannai, Shunsuke Inenaga, and Dominik Köppl. Computing all distinct squares in linear time for integer alphabets. In *CPM*, pages 22:1–22:18, 2017. doi:10.4230/LIPICS.CPM.2017.22.

**21** Hideo Bannai, Shunsuke Inenaga, and Dominik Köppl. Computing all distinct squares in linear time for integer alphabets. In *28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017*, volume 78 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CPM.2017.22.

**22** Jean Berstel and Dominique Perrin. The origins of combinatorics on words. *Eur. J. Comb.*, 28(3):996–1022, 2007. doi:10.1016/J.EJC.2005.07.019.

**23** Richard S. Bird. Two dimensional pattern matching. *Inf. Process. Lett.*, 6(5):168–170, 1977. doi:10.1016/0020-0190(77)90017-5.

**24** S. Brlek and S. Li. On the number of squares in a finite word. *arXiv*, 2022. URL: http://arxiv.org/abs/2204.10204.

**25** Srecko Brlek and Shuo Li. On the number of distinct squares in finite sequences: Some old and new results. In *Combinatorics on Words - 14th International Conference, WORDS 2023*, pages 35–44, 2023. doi:10.1007/978-3-031-33180-0_3.

**26** Gerth Stølting Brodal, Rune B. Lyngsø, Anna Östlin, and Christian N. S. Pedersen. Solving the string statistics problem in time o(n log n). In *ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 728–739. Springer, 2002.

**27** P. Charalampopoulos, J. Radoszewski, W. Rytter, T. Waleń, and W. Zuba. The number of repetitions in 2D-strings. In *28th Annual European Symposium on Algorithms, ESA 2020*, pages 1–18, 2020. doi:10.4230/LIPICS.ESA.2020.32.

**28** Ying Choi and Tak Wah Lam. Dynamic suffix tree and two-dimensional texts management. *Inf. Process. Lett.*, 61(4):213–220, 1997. doi:10.1016/S0020-0190(97)00018-5.

**29** Raphaël Clifford, Allyx Fontaine, Tatiana Starikovskaya, and Hjalte Wedel Vildhøj. Dynamic and approximate pattern matching in 2D. In *SPIRE*, pages 133–144, 2016. doi:10.1007/978-3-319-46049-9_13.

**30** Maxime Crochemore. An optimal algorithm for computing the repetitions in a word. *Information Processing Letters*, 12(5):244–250, 1981. doi:10.1016/0020-0190(81)90024-7.

**31** Maxime Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, 12(5):244–250, 1981. doi:10.1016/0020-0190(81)90024-7.

**32** Maxime Crochemore, Leszek Gasieniec, Ramesh Hariharan, S. Muthukrishnan, and Wojciech Rytter. A constant time optimal parallel algorithm for two-dimensional pattern matching. *SIAM J. Comput.*, 27(3):668–681, 1998. doi:10.1137/S0097539795280068.

**33** Maxime Crochemore, Leszek Gasieniec, Wojciech Plandowski, and Wojciech Rytter. Two-dimensional pattern matching in linear time and small space. In *STACS 95, 12th Annual Symposium on Theoretical Aspects of Computer Science*, pages 181–192, 1995. doi:10.1007/3-540-59042-0_72.

**34** Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithms on strings*. Cambridge University Press, 2007.

**35** Maxime Crochemore and Lucian Ilie. Maximal repetitions in strings. *J. Comput. Syst. Sci.*, 74(5):796–807, 2008. doi:10.1016/j.jcss.2007.09.003.

**36** Maxime Crochemore, Lucian Ilie, and Liviu Tinta. The "runs" conjecture. *Theor. Comput. Sci.*, 412(27):2931–2941, 2011. doi:10.1016/j.tcs.2010.06.019.

37 Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Jakub Radoszewski, Wojciech Rytter, and Tomasz Walen. Extracting powers and periods in a word from its runs structure. *Theor. Comput. Sci.*, 521:29–41, 2014. doi:10.1016/J.TCS.2013.11.018.

38 Maxime Crochemore and Wojciech Rytter. Efficient parallel algorithms to test square-freeness and factorize strings. *Inf. Process. Lett.*, 38(2):57–60, 1991. doi:10.1016/0020-0190(91)90223-5.

39 Maxime Crochemore and Wojciech Rytter. Usefulness of the Karp-Miller-Rosenberg algorithm in parallel computations on strings and arrays. *Theor. Comput. Sci.*, 88(1):59–82, 1991. doi:10.1016/0304-3975(91)90073-B.

40 Maxime Crochemore and Wojciech Rytter. On linear-time alphabet-independent 2-dimensional pattern matching. In *LATIN '95: Theoretical Informatics*, pages 220–229, 1995. doi:10.1007/3-540-59175-3_91.

41 Maxime Crochemore and Wojciech Rytter. Squares, cubes, and time-space efficient string searching. *Algorithmica*, 13(5):405–425, 1995. doi:10.1007/BF01190846.

42 A. Deza, F. Franek, and A. Thierry. How many double squares can a string contain? *Discrete Applied Mathematics*, 180:52–69, 2015. doi:10.1016/J.DAM.2014.08.016.

43 R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950. URL: http://www.jstor.org/stable/1969503.

44 Jonas Ellert and Johannes Fischer. Linear Time Runs Over General Ordered Alphabets. *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, pages 63:1–63:16, 2021. doi:10.4230/LIPICS.ICALP.2021.63.

45 Jonas Ellert, Pawel Gawrychowski, and Garance Gourdel. Optimal square detection over general alphabets. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 5220–5242. SIAM, 2023. doi:10.1137/1.9781611977554.CH189.

46 Nathan J. Fine and Herbert S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. doi:10.2307/2034009.

47 Aviezri S. Fraenkel and Jamie Simpson. How many squares can a string contain? *Journal of Combinatorial Theory, Series A*, 82(1):112–120, 1998. doi:10.1006/jcta.1997.2843.

48 Zvi Galil and Kunsoo Park. Alphabet-independent two-dimensional witness computation. *SIAM J. Comput.*, 25(5):907–935, 1996. doi:10.1137/S0097539792241941.

49 P. Gawrychowski, S. Ghazawi, and Gad M. Landau. Lower bounds for the number of repetitions in 2D strings. In *SPIRE 2021*, pages 179–192, 2021. doi:10.1007/978-3-030-86692-1_15.

50 Raffaele Giancarlo. A generalization of the suffix tree to square matrices, with applications. *SIAM J. Comput.*, 24(3):520–562, 1995. doi:10.1137/S0097539792231982.

51 Raffaele Giancarlo and Roberto Grossi. On the construction of classes of suffix trees for square matrices: Algorithms and applications. *Inf. Comput.*, 130(2):151–182, 1996. doi:10.1006/INCO.1996.0087.

52 Mathieu Giraud. Not so many runs in strings. In *Language and Automata Theory and Applications, Second International Conference, LATA 2008*, volume 5196, pages 232–239. Springer, 2008. doi:10.1007/978-3-540-88282-4_22.

53 Mathieu Giraud. Asymptotic behavior of the numbers of runs and microruns. *Inf. Comput.*, 207(11):1221–1228, 2009. doi:10.1016/j.ic.2009.02.007.

54 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

55 Dan Gusfield and Jens Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. *J. Comput. Syst. Sci.*, 69(4):525–546, 2004. doi:10.1016/J.JCSS.2004.03.004.

56 Jin-Ju Hong and Gen-Huey Chen. Efficient on-line repetition detection. *Theor. Comput. Sci.*, 407(1-3):554–563, 2008. doi:10.1016/j.tcs.2008.08.038.

57 Ramana M. Idury and Alejandro A. Schäffer. Multiple matching of rectangular patterns. *Inf. Comput.*, 117(1):78–90, 1995. doi:10.1006/INCO.1995.1030.

**58** L. Ilie. A simple proof that a word of length $n$ has at most $2n$ distinct squares. *Journal of Combinatorial Theory, Series A*, 112(1):163–164, 2005. doi:10.1016/J.JCTA.2005.01.006.

**59** L. Ilie. A note on the number of squares in a word. *Theoretical Computer Science*, 380(3):373–376, 2007. doi:10.1016/J.TCS.2007.03.025.

**60** Juha Kärkkäinen and Esko Ukkonen. Two- and higher-dimensional pattern matching in optimal expected time. *SIAM J. Comput.*, 29(2):571–589, 1999. doi:10.1137/S0097539794275872.

**61** Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977. doi:10.1137/0206024.

**62** Tomasz Kociumaka, Solon P. Pissis, Jakub Radoszewski, Wojciech Rytter, and Tomasz Walen. Fast algorithm for partial covers in words. *Algorithmica*, 73(1):217–233, 2015.

**63** Roman M. Kolpakov and Gregory Kucherov. Finding maximal repetitions in a word in linear time. In *40th Annual Symposium on Foundations of Computer Science, FOCS 1999*, pages 596–604. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814634.

**64** Dmitry Kosolobov. Online square detection. *CoRR*, abs/1411.2022, 2014. arXiv:1411.2022.

**65** Dmitry Kosolobov. Online detection of repetitions with backtracking. In *Combinatorial Pattern Matching - 26th Annual Symposium, CPM 2015*, volume 9133, pages 295–306. Springer, 2015. doi:10.1007/978-3-319-19929-0_25.

**66** N. H. Lam. On the number of squares in a string. *AdvOL-Report 2*, 2013.

**67** Ho-fung Leung, Zeshan Peng, and Hing-Fung Ting. An efficient algorithm for online square detection. *Theor. Comput. Sci.*, 363(1):69–75, 2006. doi:10.1016/J.TCS.2006.06.011.

**68** Michael G. Main and Richard J. Lorentz. An $\mathcal{O}(n \log n)$ algorithm for finding all repetitions in a string. *J. Algorithms*, 5(3):422–432, 1984. doi:10.1016/0196-6774(84)90021-X.

**69** Adam Marcus and Gábor Tardos. Excluded permutation matrices and the stanley-wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004. doi:10.1016/J.JCTA.2004.04.002.

**70** Shoshana Neuburger and Dina Sokol. Succinct 2D dictionary matching. *Algorithmica*, 65(3):662–684, 2013. doi:10.1007/S00453-012-9615-9.

**71** Simon J. Puglisi, Jamie Simpson, and William F. Smyth. How many runs can a string contain? *Theor. Comput. Sci.*, 401(1-3):165–171, 2008. doi:10.1016/J.TCS.2008.04.020.

**72** Jakub Radoszewski. Linear time construction of cover suffix tree and applications. In *ESA*, volume 274 of *LIPIcs*, pages 89:1–89:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

**73** Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing: Volume 1 and 2*. Computer Science and Applied Mathematics. Academic Press, Orlando, FL, 2 edition, 1982.

**74** Wojciech Rytter. The number of runs in a string: Improved analysis of the linear upper bound. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 184–195, 2006. doi:10.1007/11672142_14.

**75** A. Thierry. A proof that a word of length $n$ has less than $1.5n$ distinct squares. *arXiv*, 2020. URL: http://arxiv.org/abs/2001.02996.

**76** A. Thue. Über unendliche Zeichenreihen. *Norske Vid. Selsk. Skr., I Mat.–Nat. Kl., Christiania*, 7:1–22, 1906.